Sahaj Computer Solutions

# General Purpose Utilities

Chapter 3

Unit I

# What will you learn

- Display calendar using the **cal** command.
- Display current system date and time using **date** command.
- Use **echo** with escape sequences to display a message on the terminal.
- Use **bc** as a calculator with a decimal, octal or hexadecimal  number as the base.

# What will you learn

- The basics of electronic mail and its addressing scheme.
- Handling your mail with character –based **mailx** program.
- Change your password with **passwd**.
- Displaying the list of  users currently working on the system with who.
- Displaying the characteristics of your operating system with **uname**.
- **tty and stty**.

# cal: The calendar

- The cal command is used to invoke the calendar of any specific month, or a complete year.
- Synopsis:

  cal [ [month]year]
- Everything within rectangular brackets is optional, so cal can be used without arguments, in which case it displays the calendar of the current month.

# cal: The calendar

- To see the calendar of a specific month you need to specify two parameters with the cal command.
- Ex: to display the calendar of MARCH 2006 we can write

  $ cal 03 2006
- When we print calendar of the entire year it will not fit in the current screen page, it scrolls off too rapidly before you can use [ctrl-s] to pause it.

# cal: The calendar

- To make cal put in the same way man pauses , use cal with a pager( more or less) using the |(pipeline) symbol to connect them.
- Ex:

  $ cal 2006 | more
- The | symbol connects two commands (in pipe line) where more takes input from the cal command.

# date: Displaying the system date

- Date command displays the current date and time to the nearest second.
- Synopsis:

  $ date
- The command can also be used with suitable format specifier as arguments.
- Each format is preceded by the + symbol, followed by the % operator, and a single character describing the format.

# date: Displaying the system date

- For instance, you can print only the month using the format +%m

  $ date +%m

- Or the month name

  $ date +%h

- Or you can combine them in one command.

  $ date +”%h %m”

# date: Displaying the system date

- There are many other format specifiers:
  - d – The day of the month (1-31)
  - y – The last two digits of the year.
  - H, M and S – The hour, minute and second respectively.
  - D – The date in the format mm/dd/yy.
  - T – The time in the format hh:mm:ss.
- When you use multiple formats then you must enclose within double quotes.

# echo: Displaying a message

- Echo command is used often in shell scripts to display diagnostic messages on the terminal , or to issue prompts for taking user input.
- Originally echo was an external command but now all shells have echo built in.
- An escape sequence is generally two character string beginning with a \(backslash).
- An escape sequence is used at the end of the string and is used as an argument to echo.

# echo: Displaying a message

- Example:

  $ echo “Enter Filename: \c”

- Like \c there are other escape sequences.

| Escape Sequence | Description |
|---|---|
| \t | Tab space |
| \n | New Line character |

# Escape sequence used by echo and printf

| Escape Sequence | Description |
|---|---|
| \07 | System alarm or beep |
| \c | No new line (cursor in the sameline) |
| \f | Form feed |
| \r | Carriage return |
| \v | Vertical tab |
| \\ | Backslash |
| \a | Bell |

# printf: an alternative for echo

- The printf command is available in unix and is an alternative for echo.
- Like echo it is an external command
- Synopsis:

  $ printf “No file name entered \n”
- Printf uses formatted strings in the same way as C language uses.
- Here are commonly used formatted string

# printf : an alternative for echo

| Formatted String | Description |
|---|---|
| %s | String |
| %30s | As above but printed in a space 30 character wide |
| %d | Decimal integer |
| %6d | As above but printed in a space 6 character wide |
| %o | Octal Integer |
| %x | Hexadecimal integer |
| %f | Floating Point Number |

# bc: The Calculator

- Unix provides two types of calculators
  - A graphical  object using the **xcalc** command .
  - And the text-based bc command.
- The former is available in X windows and is easy to use.
- The other one is  less friendly, exteremly powerful  and remains one of the system's neglected tools.

# bc: The Calculator

- To use bc use the following command
- Synopsis:

  $ bc
- Example:

  $ bc

  12 *12

# bc: The Calculator

- bc can take multiple inputs each separated by a ;
- Example:

```
$ bc
12 *12 ; 2 ^3
```

Output:

```
144
8
```

# bc: The Calculator

- bc can perform only integer computations and truncates the decimal portion that it sees.
- For example : 9/5 will produce 1 as output.
- To enable floating point computations, you have to set scale to the number of digits of precision before you key in the expression.

- For example:

  Scale=2 *truncates to 2 decimal places*

  *17/7*

  *2.24*

- bc has another use and that is converting numbers from one base to another.
- Set ibase(input base) to 2 before you provide the number

# bc: The calculator

- Example:

  ibase=2

  11001010

  202

- The reverse is alsopossible

  obase=2

  14

  1110

- Bc also comes with library that can perform scientific calculations.

# Script: Recording your session

- Script command, virtually lets the unix users to record their session in a file.
- Synopsis:

```
$ script
        Script started file is typescript
$ exit
        Script stopped file is typescript
```

# Email Basics

- A unix system is used by multiple users, so communication through the system seems natural and necessary.
- Its no wonder that email is the first application that unix users are familiar with.
- An email never appears on your terminal the moment it is received.

# Email basics

- It is deposited in your mailbox even when you are not logged in.
- The shell regularly checks this  mailbox, and when detects the arrival of new mail, it issues a message to the user.

# Mail addressing scheme

- A recipient is identified by her email address.
- The addressing scheme in early days simply used the recipient's username as the email address like

  mailx  henry
- Or  a combination of username and machine name (called the host name) like

  mailx henry#saturn

# Mailx: the universal mailer

- Mailx is character based mailing agent in unix.
- Mailx finds the place in the POSIX specifications.
- There are two ways of invoking mailx—In the sending and receiving modes
- In sending mode mailx is usually used with the email address of the recipient’s address as an argument.

# Mailx: universal mailer

- In receiving mode you use mailx without arguments to handle your received mails.
- In this mode you can perform all mail functions.

# Sending Mail

- In the sending mode, mailx turns interactive, prompting for the subject first.
- You have to key in the subject before entering the message body.
- This is how henry send mail to charlie

**$ mailx charlie**

Subject: **New System**

**The new system will start functioning from next month.**
**Convert your files by next week-henry**

[ctrl+d]

EOT indicates End of Text

# Sending mail

- Sending mail is simple as the above example.
- The sent message doesn’t directly appear on charlie’s terminal but lands in his mailbox, which is usually /var/mail/charlie.
- To send the mail noninteractively use the following command:

mailx –s “New System” charlie < message.txt

# Sending mail

- Since we need to send mail through a shell script, we use a shell feature called redirection to take message from a file and the –s option to specify the subject;
- Mailx can also send copies of mail to other users by using the option –c
- Example:

```
Mailx –s “New Computer System” –c “jim , sumit” charlie
<message.txt
```

# Sending mail

- The above command will send copies of the mail to jim and sumit.
- Note: if a command line is placed in the shell, mail will be sent without user intervention.

# Receiving Mail

- All incoming mail is appended to the mailbox.
- This is a textfile named after the user-id of the recipient.
- Unix maintains the mailbox  in the directory which is usually /var/mail
- Charlie’s mail is saved in /var/mail/charlie

# Recieving

- When charlie logs in he must issue  mailx command in the receiving mode.
- Example: $ mailx
- The system first shows the  headers and some credentials of all incoming mail that’s still held in the mailbox.
- To view any message charlie has to either use the message number or press enter on the current mail , which is marked by > pointer.

# Mailx Internal Commands

- Like any other commands in unix the mailx command supports a number of internal commands. Enter ? Or help at this prompt to see the the entire list
  - Replying a mail: the r command enables the recipient to reply when the sent message is on display on the terminal.
  - Saving Message: With w command , you can save one or more messages in separate files.

# Mailx Internal Commands

| Command | Action |
|---|---|
| + | Prints next message |
| - | Prints previous Message |
| N | Prints message numbered N |
| h | Prints headers of all messages |
| d N | Deletes message N |
| u N | Undeletes message N |
| s flname | Saves current message with headers in flname |

# Mailx Internal Commands

| Command | Description |
|---|---|
| w flname | Saves current message without headers in flanme |
| m user | Forwards mail to user. |
| r N | Replies to sender of the message N |
| q | Quits mailx program |
| ! cmd | Runs unix command cmd |

# passwd: Changing your password

- To change the password  of any unix user use the passwd command.
- Synopsis

  $ passwd
- passwd expects you to respond three times.
  - First it  prompts for the old password.
  - Next it checks whether you have entered a valid password, and if you have , it then prompts for the new password.

# passwd: Changing your password

- Enter the new password with password naming rules applicable to your system.
- Finally , passwd asks you to re enter the new password.
- If everything goes smoothly , the new password is registered by the system.
- When you enter the password , the string is encrypted by the system.

# Password Framing rules and Discipline

- There are some rules that you are expected to follow when handling your own password
  - Don’t choose a password similar to old one.
  - Don’t use commonly used names like names of friends, relatives, pets and so forth. A system may check its own directory and throw out those passwords that are easily guessed.

# Password Framing rules and Discipline

- Use a mix of alphabetic and numeric characters. Enterprise unix don’t allow passwords that are wholly alphabetic or numeric
- Do not write the password in an easily accessible document.
- Change the password regularly.

# who: Who are the users?

- Unix maintains accounts of all users who are logged on to the system.
- It often a good idea to know their user-ids so you can mail them messages.
- The who command displays an informative listing of these users:
- Synopsis:

```
$ who
```

# who: Who are the users?

| Username | Device names and terminals | Date | Time | Machine name |
|---|---|---|---|---|
| root | Console | Aug 1 | 07:51 | (: 0) |
| kumar | pts/10 | Aug 1 | 07:56 | (pc123.heavens.com) |
| sharma | pts/6 | Aug 1 | 02:10 | (pc125.heavens.com) |
| | | | | |

# who: Who are the users?

- The first column shows the usernames working on the system.
- The second column shows the device names of their respective terminals.
- The third , fourth and fifth columns show date and time of logging in.
- The last column shows the machine name from where the user logged in.

# who: Who are the users?

- Most unix commands to avoid cluttering the display with header information, this command does have a header option (-H).
- This option prints the column headers, and when combined with –u option, provides a more detailed list.
- To know specifically , who is currently logged in:

```
$ who am i
```

# uname: Knowing your machine's characteristics

- The uname command displays certain features of the operating system running on your machine.by default it simply displays the name of the operating system
- Synopsis: $ uname
- The Current Release(-r) Since unix comes in various flavours, to know know which version of os you are using , you can make use of –r command
- Synospsis: $ uname -r

# uname: Knowing your machine's characteristics

- If you are connected to internet then the hostname is your machine name.
- To know the machine name user –n option with uname.
- Synopsis:

```
$ uname -n
```

# tty: Knowing your terminal

- Unix treats even terminal as files.
- The tty(teletype command ) lists the terminal the user is working on.
- Synopsis:

  $ tty

- You can control the behavior of the script depending on the terminal it is invoked from.

# stty: Displaying and setting terminal characteristics

- Different terminals have different characteristics and your terminal may not behave in the way you expect.
- The stty command helps straighten these things out; it both displays and changes the settings.
- The –a (all) option displays the current settings.
- Synopsis: **$ stty –a**

# stty: Displaying and setting terminal characteristics

- stty considers very large number of keywords, which are in the form
  - Keyword =value
  - keyword or –keyword. The – prefix implies that the option is turned off.

# Changing the settings

- Whether backspacing should erase a character:
  - If you have worked on various terminals you would have noticed that backspacing over a character sometimes removes it from sight or sometimes doesn't
  - This is decided by the keyword echoe
  - Synopsis

    $ stty -echoe

# Changing the settings

- Changing a password through a Shell Script (echo)
  - The echo setting has to be manipulated to let shell programs accept a password like string that must not be displayed on the screen.
  - By default the option is turned on, but you can turn off by

    $ stty -echo

# Changing the settings

- Changing the interrupt key(intr)
  - Stty also sets the functions for some of the keys.
  - For instance if you like to use [ctrl –c] as the interrupt key instead of [delete], use the following command

    $ stty intr\^c

# Changing the settings

- Changing End-Of –File  key (eof)
  - Usually we use [ctrl-d] to terminate the input.
  - This eof character is also selectable.
  - Instead of using [ctrl-d] we can use [ctrl-a] as the eof characters
  - Synopsis

    stty  eof\^a

# Changing the settings

- When everything else fails (sane)
  - stty also provides another argument to set the terminal characteristics to value that will work on most terminals.
  - Use the word sane as a single argument to the command.

    stty sane